BACHELOR DEGREE THESIS

TITLE: Game of questionnaires in a gamification application

DEGREE: Bachelor's degree in Telecommunication Systems Engineering

AUTHOR: David Hernández López

ADVISORS: Miguel Valero García
Roc Messeguer Pallarès

DATE: 01 of July, 2019

Title: Game of questionnaires in a gamification application

Author: David Hernández López

Advisors: Miguel Valero García
Roc Messeguer Pallarès

Date: 01 of July, 2019

Overview

This final project has focused on the improvement of an existing application called Classpip, which has been developed by several students of this faculty (EETAC). This application has as its main objective the gamification, which consists of using the techniques and elements of the games (points, objectives ...) to promote and increase the learning of the students using technology.

For this, some predefined languages were chosen and an architecture was created, which consists of a mobile application for the student to play, with an administration panel for the teacher, in which they can configure the different modules of the application. These two blocks are fed from a database based on REST-API.

The main objectives of this project can be defined as:

- Modify and improve a current first version of a questionnaire module, which does not have great functionality or playability for students. That is why I have the freeway to develop, implement or modify any functionality.

- Provide help to the future developers. These aids consist of, from helping them understand my code by providing them with a video tutorial where I talk about it, to help them to be productive as soon as possible, making an example of how a functionality could be implemented in the application.

Finally, a much more improved version has been achieved than the current one, which has many open lines because there are configured parameters that are not used. Therefore, the open lines can be considered to be started and marked.

Título: Juego de cuestionarios en una aplicación de ludificación.

Autor: David Hernández López

Directores: Miguel Valero García
Roc Messeguer Pallarès

Fecha: 01 de Julio, 2019

Resumen

Este proyecto de final de grado se ha centrado en la mejora de una aplicación ya existente llamada Classpip, la cual ha sido desarrollada por varios alumnos de esta facultad (EETAC). Esta aplicación tiene como principal objetivo la ludificación, la cual consiste en usar las técnicas y elementos de los juegos (puntos, objetivos...) para fomentar y potenciar el aprendizaje de los alumnos mediante el uso de la tecnología.

Para ello, se parte de unos lenguajes y una arquitectura ya predefinidos, la cual consiste en una aplicación móvil, para que el alumno juegue, con un panel de administración para el profesor, en el cual podrá configurar los diversos módulos de la aplicación. Estos dos bloques se alimentan de una base de datos el cual se basa en REST-API.

Los principales objetivos de este proyecto se pueden definir como:

- Modificar y mejorar una actual primera versión del modelo de cuestionario, el cuál no tiene grandes funcionalidades ni jugabilidad para los alumnos. Es por ello que tengo vía libre para desarrollar, implementar o modificar cualquier funcionalidad.
- Dotar a futuros desarrolladores de ayudas. Estas ayudas consisten en, desde ayudarles a entender mi código proporcionándoles un video tutorial donde hablo sobre ello, hasta una ayuda para que puedan ser productivos cuanto antes, haciendo un ejemplo de cómo se implementaría una funcionalidad a la aplicación.

Finalmente se ha conseguido una versión mucho más mejorada que la actual, la cual tiene muchas líneas abiertas debido que hay parámetros configurados que no se usan en la actualidad. Por ello, las líneas abiertas se pueden considerar que están empezadas y marcadas.

Acknowledgements

Thanks to my advisor Miguel, for supporting me
and advising me on any questions I had
related to the project Classpip.

To my parents, my sister, my girlfriend and my whole family,
thanks to whom I am who I am
and to whom I can only express my sincere gratitude
for supporting me during these bachelor degree that today ends.

CONTENT

CHAPTER 1. INTRODUCTION

In this first chapter, I am going to introduce the main ideas of the entire project. Starting from the main objectives of the Classpip project, the reason why I want to be part of this project explaining my personal motivation and, finally, how this document is structured.

1.1. Classpip project objectives

Nowadays, technology is part of any aspect of anyone's everyday life. In contrast, in the educational field, technology is not taking effect as quickly as in other sectors, since there are many people who offer inconveniences to change the teaching methodology. Despite this, the concept of Gamification is strongly committed to breaking these barriers. This project called Classpip focuses on that.

Another important factor in which Gamification wants to bet, is the lack of motivation of many students. It is easy for a child lose their motivation in a class, which becomes boring because there is nothing more to do than reading and writing every day. Another important factor is that each student needs a different kind of help and learn in a different rhythm. In addition, each of them is interested in different topics that could not be all related to mathematics, science or history. Thanks to gamification, we can create some specific paths for each of them.

Classpip wants to use technology and the gamification methodology to enter the classrooms and combat that lack of motivation of many students. The gamification methodology uses mechanics and techniques found in games for students to learn while having fun in them. The Computer Architecture Department[1] created this project and several students are developing the application. The teacher will use the application on the computer (WebApp), while the students can use it on their smartphones or tablets.

The application is divided in three big blocks: Services, Dashboard and Mobile. The application consists of a set of modules (**Figure 1.1.**) connected between them that make the application very dynamic and useful. For each module, we have two different options depending on which is the role of the user: student or teacher. Some modules are not available for students.

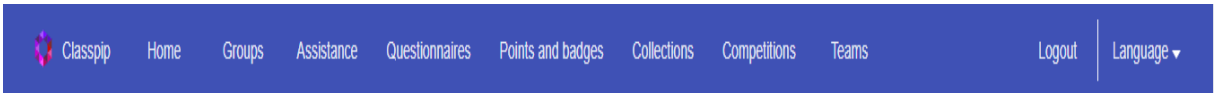


Figure 1.1. Navbar of the application Classpip (Dashboard)

[1] The Department of Computer Architecture (DAC) of the *Universitat Politècnica de Catalunya* (UPC) teaches and promotes research in topics related to the fields of Computer Architecture, Computer Networks and Operating Systems.

Students have the following modules:

- Home: Here they can see some information about them. The number of points, the ranking of their class, the level they have, to which school they belong…
- Groups: Here they can only see which group they belong to, but they do not have access of which members make the group.
- Points and badges: Here they can view his points, badges and the ranking of their group.

Teachers have the following modules:

- Home: Here they can see some information about them. Their name, their address, the student's ranking…
- Groups: Here they can manage the groups, according to the class they belong to. They can also view all the students of each group.
- Assistance: Here they can roll call choosing the group.
- Questionnaires: Here they can create and delete questionnaires.
- Points and badges: Here they can assign or take points and badges to students or teams. They can also create new points or badges. Finally, they can view a ranking of the students, sorted by number of points.
- Collections: Here they can assign collections to a team, or cards to a member. They can also view all active collections and create new ones.
- Competitions: They can create, view and delete competitions, assigning everything they want: rewards, teams, groups, etc.…
- Teams: They can create, view and delete teams of each group they have.

As you can see, the part of the dashboard is not entirely useful for the student because most of the modules are only available to the teacher. It could be considered to delete it or making it more useful.

There is also another extra role in the application, which is "School admin". This is a special role, because they have only access to a specific part of the configuration of the schools. As responsible of the School, they can add, edit or delete new schools or faculties. They cannot do anything with the other modules, since they are for the role of the teacher or the student.

1.2. Project outline

Having now a global vision of the project, the main objective of this thesis and my contribution to this great project will be to develop a module of questionnaires much more powerful and much more useful than the current one, with the implementation of different types of games. The idea is making it simple to use but at the same time with many types of possible configurations. To do this, I will first brainstorm by seeing and testing current applications in the market, and then design what I want to implement based on the best ideas of each of them.

The main idea is to design games, in this case game of questionnaires. A game of questionnaires is created by the teacher and assigned to a group class. It consists on fulfilling several related questions, and obtaining a final grade according to the number of correct answers. Whoever guesses the most, will have the best score and therefore, will win the game.

Then the goal is, how to make as attractive as possible the way that how questionnaires are fulfilled by the students, without forgetting the teacher's facility to create and reuse them. Then students will be able to play while they are learning and having fun using questionnaires.

Others objectives of this project are to make more tutorials for future developers and also a tutorial of how to use my modules for the teachers, get real uses of our application talking with teachers and other little objectives that are related to these objectives.

The structure of the project can be summarized as follows:

First of all, a main view of all the main things of the bachelor degree thesis:

- An explanation of what gamification is and how Classpip is using it can be found on **Chapter 2**. We will also take a gander real applications that are available in the market nowadays, which will give us the best ideas to develop our module. This ideas will be part of our brainstorming.
- On **Chapter 3** we can find the introduction of the project, where we will talk about how the structure of the Classpip application is since it was created and what modules are already implemented. Finally which will be my contribution to Classpip.
- A top view of what I want to implement in the project can be found on **Chapter 4**, where I talk about all the possibilities I gaze in the design. There will introduce the concept game that is new. Also a few sketches of possible screens designs of the application, thinking about the easiest ways to move between screens and where to display the database (DB) information.
- On **Chapter 5**, it is explained how I manage to make all the possible designs, explaining which I am able to develop. Finally which were the results and which parts can be easily improve, taking into account some fields from the database that are not used.

Secondly, other parts of the project that are no less important, only there are not the main objective of the thesis:

- On **Chapter 6**, there is some own and external test and assessments of the application, trying to assure that everything works correctly in the application.
- OnBoarding materials for future developers, explaining how my module is already implemented, taking them to a walk around the code. Also a short video showing them, how to add some new feature on the application. These will be implement, doing some video tutorials, which can be found on **Chapter 7**.

> Finally, on **Chapter 8**, the conclusion of the thesis, talking about some technical points, give some personal feedback about my feelings developing the application and give some details of open lines about the project.

1.3. Personal motivation

During all the years of my university career, I have experienced several changes in professional/work goals. I joined the degree desiring to know everything about telecommunication systems, from how they work inside, to how we could connect with a person in the other part of the world.

After overcoming the different academic courses, I realized that the curiosity I had, had not become a work passion. That is why, due to this frustration, I had to look for other professional opportunities. Luckily, this career is multidisciplinary so it offers you many other possibilities:

- Developer in various programming languages. Along the career I have done several compulsory subjects (21 ECTS) and during my Erasmus I have done several electives (14 ECTS), which represents 15% of the credits of the whole career.
- Information Technology (IT) departments.
- Consulting/Audit technological departments.
- Robotics.
- Drones.

Given these multiple possibilities, I have been able to experience some thanks to the possibility of doing internships in multinational companies. The only one that I lack is in an application development company. That is why I want to take this chance to make a high-quality application and see if I am really capable and if I imagine doing this for the rest of my life. This big project I believe that fits this desire I have, to prove myself if I could do it.

Another personal motivation is that my goal, since I started this career was to do a bachelor degree thesis which could reach beyond the academic world, something that could have a small impact on someone outside of my circle of contacts (teachers, university, family…). These expectations have been fulfilled, since in this project there are many more people involved than the circle mentioned above, because it will finally reach an unknown customers or new developers.

Then I could say totally sure, that choosing this bachelor's thesis is a great choice for me and I will try to make a big effort to do great code.

Now it is time to begin explaining the important concept of GAMIFICATION.

CHAPTER 2. GAMIFICATION

Gamification is the main objective of the application Classpip. But, what is exactly gamification? In this chapter, I am going to explain what is Gamification, why is a great solution in education and how is used nowadays in the educational environment.

2.1. What is Gamification?

The easiest way to explain what gamification is, is going to one of the best definition that have ever been done. The most generic and precise definition of gamification could be which Gabe Zichermann[2] described in 2010:

*"Gamification is the process of using **Game Thinking** and **Game Dynamics** to **Engage Audiences** and **Solve Problems**".*

Gamification uses elements that are only considered for the games (badges, points, levels...) to make people increase the time that they will be "playing" in the game, rewarding some concrete actions. Gamification is used since many years ago and is present in the day to day of people without them realizing. Something as simple as *"If you came 4 times here to have your car washed, the fifth will be free".* The psychological behaviour will make this person want to complete and achieve the final goal (as in a game), and probably this person will get his free carwash.

If we move on to the educational field, we can define more accurate gamification as:

*"Promote **learning** by **taking advantage** of the **psychological predisposition towards the game** to **improve** the **motivation** towards learning"*

This type of learning pretends to gain more knowledge in a more fun way, making a more positive experience for the student. Using the techniques, mechanics, tools of the games that we are all familiar with, we can stimulated the students to learn things through fun.

The most usual technique used in games and adapted by the gamification are:

- The mechanic techniques which refers to the gift that the student receive when they achieve some goal:
  - Points: They are accumulate when certain actions are carried out.
  - Levels: To take into account the record of the student.
  - Gifts: Some advantages when they achieve a goal.
- The dynamic techniques which refers to the own motivation to play and achieve objectives:
  - Reward: Receive a better reward than the others students.
  - Status: Have more privileges than the others have.

[2] Gabe Zichermann is considered the world's expert and public speaker of gamification. He has several books on the subject and it is the founder of the webpage "Gamification.co".

- Competition: Win the others and try to be better than them.

The idea of Gamification is not to create a game, but is to take all this things that game has to motivate students and make them have more acknowledge, that normally is reflected on taking better marks.

2.2. Examples of applications in the education

Since this new concept, named as “Gamification” appeared, several companies are trying to develop some huge applications to help teachers to implement this concept. This are the references that Classpip had been based in order to implement a more powerful application.

In this case, I am going to explain some important applications based mainly in questionnaire. Considering these, I will try to implement a huge module of questionnaire in Classpip to be as competitive as them.

2.2.1. Kahoot

One of the most famous gamifications apps used now is *Kahoot*. (**Figure 2.1.**)

Figure 2.1. Kahoot screens

Kahoot allows the teacher to make a questionnaire with pictures and multiple answers. You can play with these questionnaires in class online: the students will join/subscribe on a questionnaire, introducing the game PIN on their smartphone/tablet. Then, on the screen appear the question and all the possible answers, which a colour related with each. Finally the marks will appear on the screen. You cannot add or modify any other feature. You can only play in the same way with your own questions.

In our case, we will try to make a similar solution of questionnaires online at class, with the improvement that the results will automatically become some benefit for the winners, because the teacher previously will configure the rewards for the winners. In addition, we will try to configure a more dynamic games, where the teacher can modify some features as their please.

2.2.2. Quizizz

Another important app, which is based on questionnaires, is *Quizizz*. (**Figure 2.2.**)

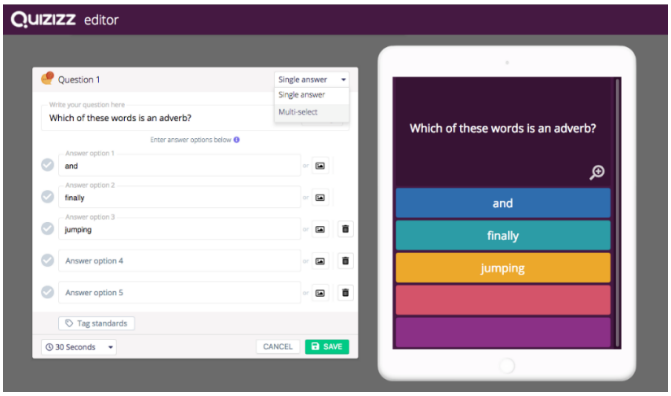


Figure 2.2. Quizizz screens (Play & configuration)

This application bets for the integration with google. All the tools from Google as Gmail, Calendar and Hangouts are implemented in this application, and it makes a powerful application: you can send emails using personalized Gmail Accounts, you can set in the group calendar the due date for the questionnaire and every child will receive a notification, reminding them that they have to make the questionnaire.

It has another amazing concept, I had not seen so far. You have the option to not tell the student his mark, you can easily tell him “Your result has improve from the last questionnaire. Keep going like that!” It can be an amazing concept for that children that normally fails the questionnaire. They could improve from a “2 mark” to a “4 mark”, but instead of telling him “You still fail the questionnaire”, you can just give him the message "You are improving!” which sounds amazing in their head.

2.2.3. PlayBrighter

Another great application, that is mostly based on questionnaires is, PlayBrighter. (**Figure 2.3.)**

Figure 2.3. PlayBrighter mission configuration

In this case, the application PlayBrighter brings the questionnaires to a game (missions). It is only focused on the science field (predefined questions), but there is the possibility to create more questions. There are different parts of a magic world, and the students can advance until they achieve the flag. They need to complete all the parts of the world and they can swap easily between them. It is easy for the teacher, know where their students are stuck, because there is a great visual world, which all the students are represented with his photography.

2.2.4. Socrative

Another great application, that is mostly based on questionnaires is, Socrative. (**Figure 2.4.**)

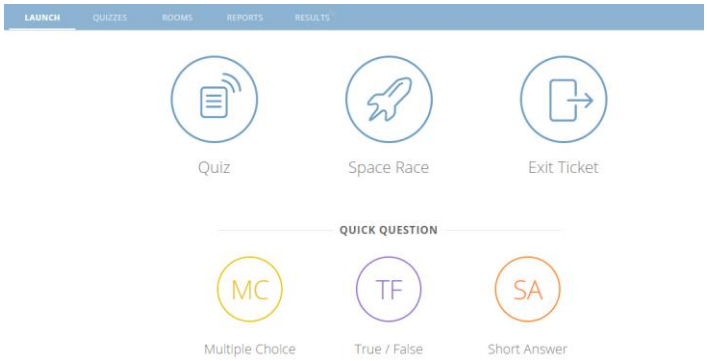


Figure 2.4. Dashboard of the teacher in Socrative

Socrative has probably the nicest design and the easiest way to do everything. Teachers have easy access to all the important parts: make games and check the results. The teacher creates the questionnaire and then launches a room, which creates a code to allow students connect to that room. There is also a team mode, where students can choose which team they belong to, and all the grades will sum up to that team, doing a “space race”. Another great feature is that teachers can share their questionnaires with another teachers.

2.3. Brainstorming

Reviewing and playing through all these applications, gives me great ideas for implementation in my module of questionnaires. These ideas could be summarized by:

- A similar game as Kahoot. Kahoot is a great application that gives students the chance to play online and compete with their teammates. Probably the best game to do it in class, thanks to the fact that competitiveness is an important factor because all of us, always want to be the best. To be the best, you must win the game, which means you must know all the answers or, in other words, study at home to win the game.
- Implementation with google tools. It would be fine, if we could implement something synchronized with Google as Google Calendar. It will be easy to remind students that they have an exam that day or that they have to make some projects for that other day. The Hangouts tool will also give students the possibility of doing projects online, speaking through their smartphones.
- Do not tell the student the actual grade if he is not passing the test yet. It might be good to avoid the grade and have the ability to say "keep improving" to the student.
- Making different paths related to different topics. It would be good that in one subject, for example in mathematics, could have many ways to go (multiplication, powers, roots…) and each student has the possibility to choose which one they do first and which one later. These could maintain the motivation of each student because each of them chooses his personal path to practices what they prefer.
- Socrative probably has the best way to implement the design of the user experience. Try to do it as simple as Socrative, which in a few steps, teacher can reach the important tools of the application. It also has easily and friendly the way to configure questions, games, launch games…

After this introduction, now it is time to move on to the project. I will start making an introduction of the project Classpip before we move deeply to my contribution.

CHAPTER 3. INTRODUCTION TO THE PROJECT

Classpip is a tool that uses gamification, especially focused on the educational field. The application architecture was developed in a TFM (Final Master Thesis), from one student in 2016, inside the AC department of the EETAC. Since then, several students have been developing some features of the application, creating this big project that we have now.

3.1. Architecture of the project Classpip

Since Classpip was created, has always keep its architecture. The only thing that has changed is that for the website (*Classpip-dashboard* project), instead of using *Bootstrap* for the web design, we use *Angular Material*, which has better features with *Angular* and has many pre-design classes for make it faster, easy and responsive our website. We can see the whole architecture in **Figure 3.1.**:

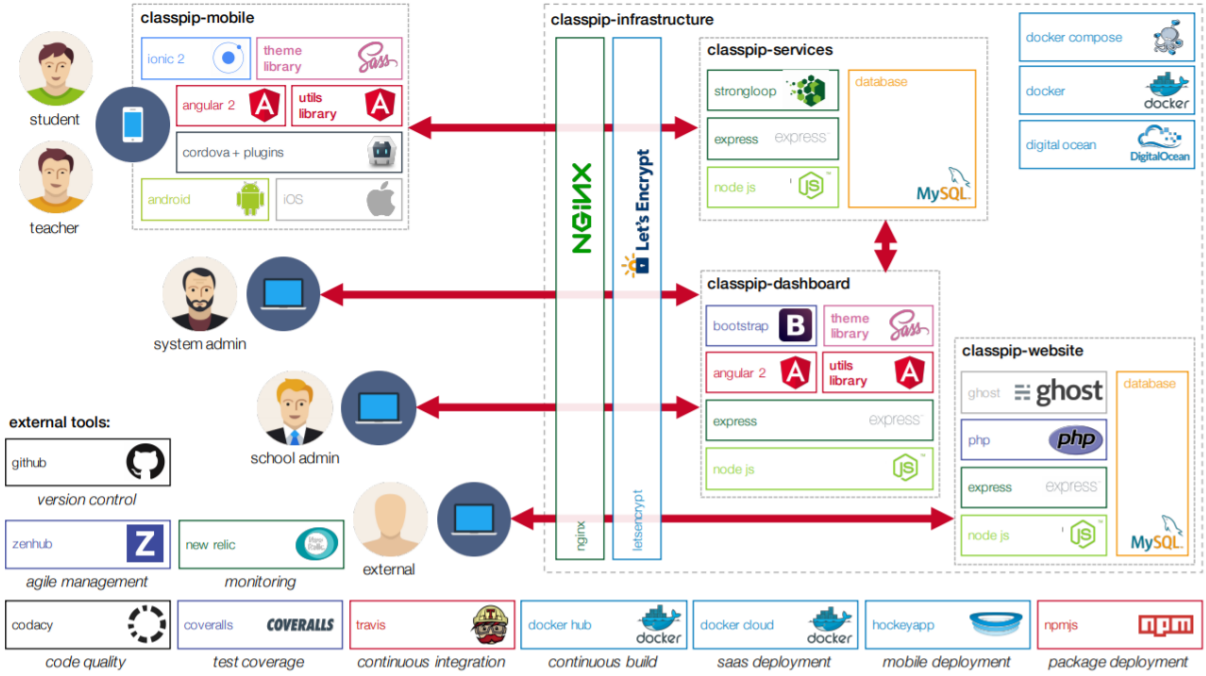


Figure 3.1. Classpip architecture

As we can see in the **Figure 3.1.**, we can identify two big groups and many external tools, which together make our big application Classpip.

The two big groups are divided in four different code repositories.

3.1.1. Classpip-Mobile

This part is focused on creating the mobile access part of the application. It is based on *Ionic 2* framework, mixing other tools like *Angular 4.4* to make it more dynamic and *Angular Material 2* for the user interface design. *Ionic* uses *Apache Cordova, which* is a powerful framework, because it can give us easy access for all the tools of the smartphones as GPS, camera, fingerprint reader…

3.1.2. Classpip-infrastructure

Which is also divided in three different repositories:

- Classpip-dashboard**:** This part is focused on the access to the application from a Web Browser, mainly from a computer. Is focused only for been used by the teacher because normally it is easily to configure all the features from a computer. It is based in *Node.js*, *Express* and *Angular* and for the user interface design *Angular Material*.

- Classpip-services: This part is focused on the database (DB). It is based in an *API REST* connected to a database. The API REST uses *Strong Loop*, which is an endpoint technology based in express and noje.js. This technology works on html web address, which our modules of *Dashboard* (WebApp) and Mobile can access to create (*PUT*), modify (*MODIFY*), check information (*GET*) or delete data (*DELETE*) in an easy way.

- Classpip-website: This part is focused on the projection of the website. Its main purpose is for publicity and marketing of the project Classpip. The blog is displayed on Internet via an *Express* web server on a *Node.js* environment. Now it has other features as: make an easy access to all the projects documents “fulfilled projects”, include all the tutorials that Classpip has, how to start developing the application…

3.1.3. Other frameworks

Other important Framework that should be explained is:

- GitHub: *GitHub* is an online repository where all the developers can upload this code in order to make collaborative project.
- Git: *Git* is a free and open source distributed version control system designed to handle everything from a programming projects. It easy to manage the versions and move forwards and backwards with efficiency. Also you can create branches to develop multiple solutions. You can upload all the versions easily to *GitHub*.

3.2. Description of the project Classpip

Classpip nowadays has several modules developed by different developers in different projects (TFG). These modules were created separately, but they were grouped together and now we have a many relationships between them that make a more powerful application. As explained in the introduction, some modules are only accessible by teachers. Unless otherwise stated, we will assume that only the teacher has access.

The different modules that are now implemented are.

3.2.1. Home

The Home page is the first screen that the user sees when they log in. This module can be accessed by two roles:

- Student: Here they can see a lot of information about them related to the day to day. They can visualize their level, taking into account the points achieved. The points and badges they have are also shown. Next, they can visualize the ranking of their class and which school they belong.

- Teacher: Here teachers can see their profile, which is the relevant information about them. They can also see the ranking of the students to see how they improve. Finally, the relevant information about their school appears, such as the street, social networks…

3.2.2. Groups

This module can be accessed by two roles:

- Student: here students can only see which groups they belong to, but they do not have access of which members belong to the group.

- Teacher: They can view all the groups (1ESO Mathematics, 3 ESO Catalan…). It is easy to quickly identify which groups they belong to. They can also access one group and review all the basic information of each student's profile: email, avatar, name....

3.2.3. Assistance

In this module, the teacher can roll call depending if the students come to class or not. This roll call will give badges to the students that arrive soon, depending on the configuration that has set the teacher.

3.2.4. Points and Badges

This module can be accessed by two roles:

- Student: Here they can see their scores according to the number of points and badges they get. They can also check the scores of their classmates.

- Teacher: Here they can create, modify and assign points and badges to all students they have. They can also assign points and badges to teams within each group. They can review the classification of each group, order by score and know exactly which points and badges each student has.

3.2.5. Collections

The teacher can create collections. Collections are basically an album sticker, a bunch of stickers that go together because they are related. After creating it, they assign one collection to a group, in order to deliver award stickers to the students. They can also delete one.

3.2.6. Competitions

Teachers can create competitions in two ways. One competition is a tennis tournament and the other a league tournament. As a tennis tournament, each student will face another student and one of them will go on to the next round, until two of them reach to the final when a student wins. The league tournament will face one student against another in each league round, until they reach the last round of the league. They can assign points and badges to the winners.

3.2.7. Teams

In this module teachers can create teams. The teams are a group of people of each group. They can create as many as they want in each group. This will allow the other modules, such as questionnaire or competition, the possibility of play in teams' mode instead of playing individually. They can also see which teams each group has.

All these modules, are mainly develop on the Dashboard. The mobile has only few of them, because most students only have time to develop the Dashboard. But as they are created and working on the Dashboard, would be easy to implement some features on the mobile repository. Another possibility is to have some modules only on the Dashboard side, while other on the mobile phone side.

3.3. Objectives of the project

There is an actual implementation that is not very efficient and friendly. In this project I have been entrusted with the task of improving or developing a new questionnaire module. Probably this is one of the most important modules of the whole project, since it is where you can compete with the rest of the students in one way or another. It is easy to make connections with the rest of the modules. For example, you can use the results of a questionnaire to decide the winners in each match in a league competition, you can award badges and points for the best scores of the questionnaire, you can give cards for the winner… It is easy to imagine many possibilities using the questionnaires results.

Another important part of the project is that the work method is collaborative. That means I have to be concern of the code I do and try to make it easy to understand. That makes the necessity to make tutorials (legible or visible) for future developers, in order facilitate the maintenance of my code.

When you develop an application, it will be used by other people. You are never one hundred percent sure if your user interface and experience design are the best suitable option. Therefore, it is advisable to attach video tutorials explaining how to use it and ask someone outside to test your application. This will make that our application improve thanks to the feedback.

Now it is time to continue with the first part of the questionnaires project, explaining how is design the module of questionnaires.

CHAPTER 4. DESIGN OF THE QUESTIONNAIRE MODULE

In this chapter, I am going to describe what procedure I follow to design the questionnaire module. I am going to describe all the different ways of playing a questionnaire game and a few sketches of a possible design.

4.1. Questionnaire module

The main idea of this module is to reproduce the typical questionnaire in multiple variations and options. From the most familiar use of questionnaires, to the most modern way of playing it. Knowing that, I brainstormed with some gamification applications (as explained above) and tried to merge them all into our application. In doing so, I ended up with several possibilities to play, from the typical questionnaire to an online questionnaire in the class, to the possibility of giving a homework questionnaire task. Therefore, the questionnaire module can easily be combined with other modules and the teacher has many possibilities to use them.

4.2. Starting point

At the first meeting with my advisor, he told me that Classpip has a very simple questionnaire module implemented. So he gives me free way to change and imagine whatever I could develop. So the first step was to analyse what could be improve:

- The structure of the DB was not the best. The concept of “questionnaire game” and “questionnaire” was the same. As you set up the questionnaire game, you are also configuring the questionnaire. Then, the databases were related by 1:1, which makes one useless and makes the idea of merging both into one the best one.
- The relation between questionnaire and question is not the best. By configuring the questionnaire (and also the questionnaire game) you are selecting and creating the questions. This relations 1:M relationship means that you cannot reuse these question in another questionnaire.
- There is also an answer database and a correct answer database, which only has an id related with one question and the id of the correct answer, respectively. This is a really useless database, because you can easily add this field to the question database.
- The possibility of multi answer is not taken into account. The only two possibilities that you can configure is and open answer or one correct answer. It would be good to add this possibility
- Each questionnaire can only have all the questions configured in open answer or just one correct answer. Then the possibilities of configuration are quite limited. It would be nice to have the possibility to add as many question as different types as you wish.

Considering all these improvements, the best methodology to follow is creating everything from scratch and only taking the original code as support. As the database is modified, how the information is access will change, so all the queries will be different.

4.3. Questionnaires games design

A questionnaire can be defined as “a set of questions that must be answered at the same time”. Knowing that, we can imagine some different ways of making games from the same questionnaire. Thanks to the initial brainstorming, we will be able to design some modern features that are working on really important gamification applications.

The initial design is that the teacher creates many independent questions. Each question can be as the teacher wish: open answer, multiple answer, multiple correct answers… After this, they create a questionnaire that is form by a set of questions (as many as the teacher wants). Each questionnaire can be design with different parameters that the teacher can configure according to their choice and the result is what we call “questionnaire game”.

All the following variations of games that I am going to explain you can have different configurations depending on what the teacher wants to play.

There are some configurations that are common for all games, which are:

- The teacher can add a deadline, which could be 30 minutes after the class starts, or they can put the questionnaire for homework which will be closed at any time they wants, for example at Sunday night.
- The teacher can configure a team mode instead of individual. Within this team mode, there are two options: the first one, each team plays the game on their smartphone/tablet. Finally, all the scores will summarized to make the final classification and each of them will get the rewards/points assigned taking into account the classification. The second one, all team members play on a single phone/tablet and only have one score. In the end, the grade will be copied to all the profiles of the members of the team, and each of them will get the rewards/points assigned considering the classification order.
- The teacher can configure if they wants to play with difficulty mode. Each question will give more or less points depending on its difficulty (easy, medium or hard).

The different type of games that I design are the following.

4.3.1. Quizpip

Standard mode, or as I named it “Quizpip”: This is the simplest and the most typical way to do a questionnaire. All questions appear on the smartphone screen, and at the end there is a “Submit” button to send all the answers. Finally, the points appears and the result of all the different questions.

In this game the teacher can set a questionnaire time, in order to establish a maximum time allowed to answer the questionnaire.

4.3.2. 1by1

Battery of questions, or as I named it “1by1”: The student answers all the questions one by one. The questions are shown one by one and the student will only have a reference of which question is answering and how many are left. There is no possibility to go back to the previous question.

In this game the teacher can set up a questionnaire time, in order to establish a maximum time allowed to answer the questionnaire. It can also set up a question time, in order to establish a maximum time allowed to answer each question

4.3.3. Qlasspip

Kahoot online, or as I named “Qlasspip”: The teacher will show the questions on the screen using the projector. Students will have the answers to click on their smartphones / tablets. After it finishes or question by question, will show the marks on the screen.

One variation of this game, could be that in the screen will only show the marks rising. The students will have also the question on his smartphones and they can answer the questions as fast as they want.

In this game the teacher can configure a questionnaire time, in order to put a maximum time allowed to answer the questionnaire. It can also configure a question time, in order to put a maximum time allowed to answer each question

4.3.4. FlipCardsPip

Cards with the question and in the backward the answer, or as I named “FlipCardsPip”: The teacher creates a questionnaire with the objective of make that their students study. Then the functionality will be a card with two sides. The first side, is shown to the students with the question but there are not any possible answer. The student will think about it, and when they think that they have the answer, they can click the card and will flip, showing the solution.

One variation of this game, could be the mode “only one time”. If we allow only one attempt for each student, they will be forced to get notes if they want to review later.

4.3.5. TimeToLearn

Random question in a random time, or as I named “TimeToLearn”: on their smartphones, a question could appear in their notification bar at any time. They will access to the application to answer it, and depending on how faster they were, the less time they take to answer, the more points they will receive. Normally the questionnaire will be set by 7 questions, one for each day of the week.

4.3.6. TrivialClip

TrivialClip is based on the actual trivial game. For winning the game, each student or team should obtain all the different colours (categories). But not all the questions are allowed to have colour and prize, a lot of questions just have for prize “another throw”. Should be create a table board in order to play. This game should be online and done in class .In this game the teacher can configure a questionnaire time, in order to put a maximum time allowed to play the game.

4.4. Database design

Trying to improve the inefficiency of the current implementation of the questionnaire database, explained above, I arrive at four important databases related between them.

First of all, let’s define exactly which the relationship between the questionnaire modules is. It can be summarize using the following picture:

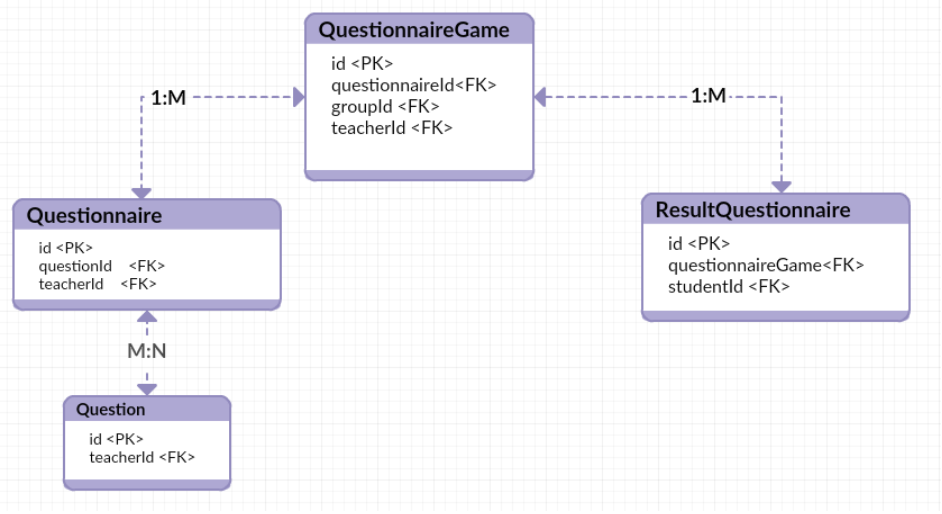


Figure 5.1. Database relationships

We only see the main fields of each database in order to know what relationships have each ono has with others databases. The relationship (X:X) between them is also explained. There are other identifiers that allow us know that they are connected to other databases that are not explained in this thesis. For example, each Questionnaire is related to a teacher (teacherId) or each questionnaireGame is related to a group (groupId).

4.4.1. QuestionnaireGame

This must be the most important one. Here we can configure some parameters like, time of question, time of questionnaire, starting date of the game… This can take several configurations.

This database will have a relation of 1:M with Questionnaires, as each game can only have one questionnaire associated.

This database will have a relation of 1:M with resultQuestionanire, since each game could have many results. To make it easier to develop, a copy of each questionnaireGame will be stored on each resultQuestionanire post.

4.4.2. Questionnaire

This database is the core for the questionnaires game. Each questionnaire will consist of a group of questions (no matter the type) whose belong together making a questionnaire.

This database will have a M:1 relationship with questionnaireGame, since each questionnaire belongs to a large number of questionnaireGame. This database will have a relation of M:N with Question, as each questionnaire can have many questions.

4.4.3. Question

This database will store all the information about questions: type of question (multiple choice, open answer…), the category of the question, the correct answer…

This database will have a M:N relationship with Questionnaire, since each question can belong to several questionnaires.

4.4.4. ResultQuestionnaire

This database will store the results of each game. It will store several parameters such as: a copy of the questionnaireGame, the student identification, the teacher identification, how many correct answers…

This database will have a relation of M:1 with questionnaireGame, since each result can only belong to one questionnaireGame.

4.5. Final design

Taking into account all this possible game design, it is time to focus on which games I will try to develop. Assuming that I will create everything from sketches (databases, modules of dashboard and mobile), it is easy to think that I will not achieve everything. Therefore, if I would like to develop something powerful and with a finish result I need to narrow the scope. Knowing that my personal contribution to this project is going to be:

- Creating the databases trying to implement fields for all the possibilities that I previously design.
- Creating the dashboard size, allowing the teacher to configure questions, questionnaires and questionnaires game as they wish. That means they will create as many questions as they want independently. Then they will create questionnaires, and after that they will create questionnaires game. This will allow to re-use both the questionnaires and the questions.
- Creating some games. The games that I will develop will be: Quizpip, FlipCardsPip and 1by1.
- Creating the mobile side, allowing the students to fulfil the questionnaires as the setup of the questionnaire says. It also includes the question/questionnaire time option.
- Save the results of the students and allow the teacher to see the results in the dashboard size.

4.6. Sketches design

After the design, the next step when you are developing some application is to create some sketches. Knowing how will be the databases, it is time to imagine how we can show all this information doing some quick sketches.

For these sketches I am going to use the *MockFlow.com* website that allows you to create quickly some sketches. It also has plugins about *Angular Material* and *Ionic* which is a good feature because you can create a good approach of your final result.

4.6.1. Dashboard

Using the Angular Material plugin of MockFlow.com we obtain the following sketches.

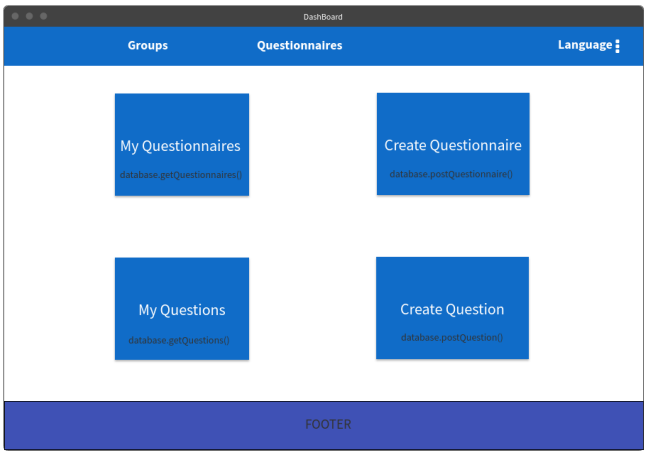


Figure 4.1. Questionnaire Home Sketch

In **Figure 4.1.** we can see which could be the *Questionnaire Home Page*. We have four buttons, each of which allows you to create or view questions and questionnaires. In this case, *My Questionnaires* and *My Questions* send an *http.get* to the database while *Create Questionnaire* and *Create Question* send an *http.post* to the database.

In **Figure 4.2.** we can see how we could see the questionnaire information. It will have a list of all the questionnaires, and when we click on one of them, we will see some relevant information about it, such as the questions it has, the answers of each questions, the image of each question… Every click will create an *http.get* in order to get the question information.

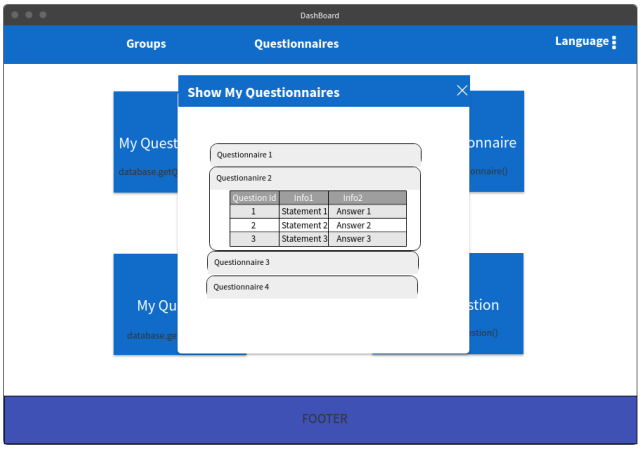


Figure 4.2. Show Questionnaires Sketch

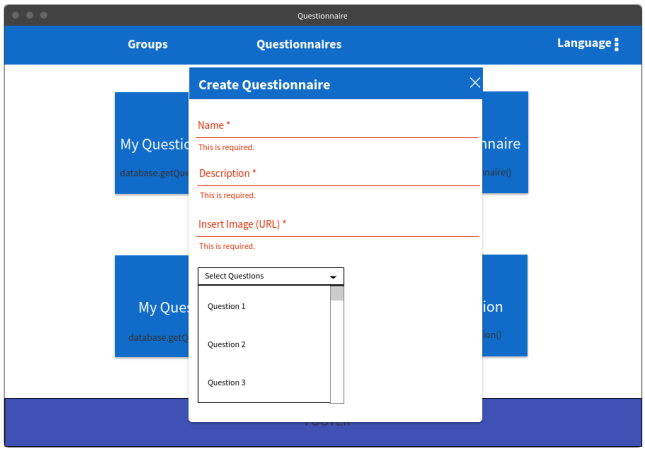


Figure 4.3. Create Questionnaire Sketch

In **Figure 4.3.** we can see how we could create a questionnaire. We can set all the parameters and, finally, select which questions we want to add to the questionnaire. As we can see, all the inputs will be required, and we could select as many question as the teacher wants. Finally there is a *Submit* button which makes the *http.post* to the database.

In **Figure 4.4.** we can see which could be the game *Home Page*. There are two buttons, one to create a new game, which sends an *http.post*, and another to see the results of the game, which sends an *http.get*. Below there is a list of all the games. It would be good to distinguish between active and non-active. Each game will show the most relevant information about it.

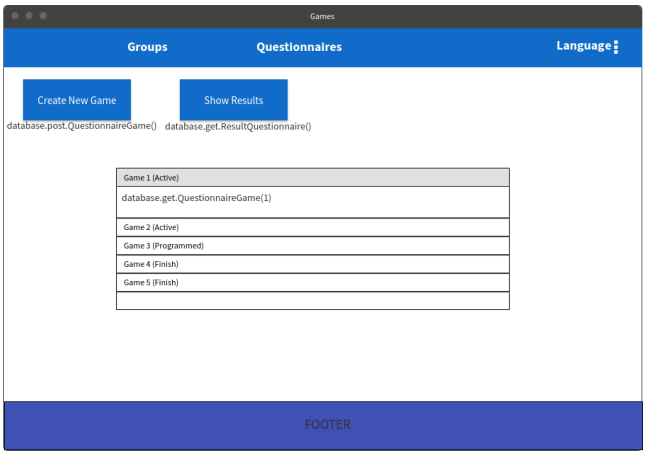


Figure 4.4. Game Home Sketch

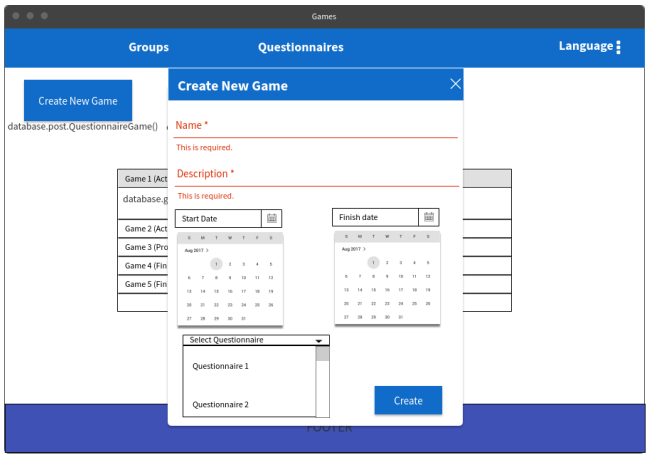
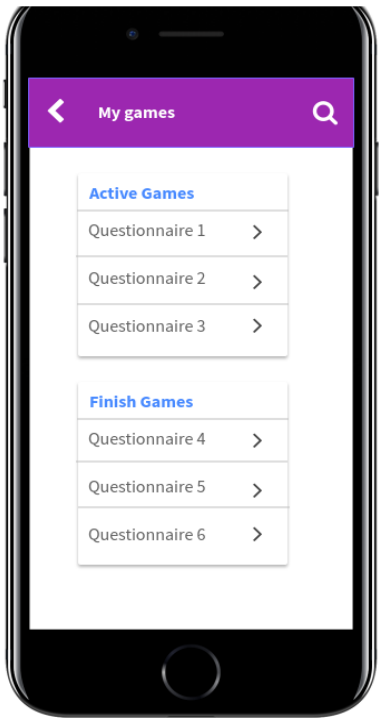


Figure 4.5. Create New Game Sketch

In **Figure 4.5.** we can see which could be how we create a new game. The games will have a start and finish date and a questionnaire related to them. It would be good to use this date information to establish active and non-active questionnaires. Finally there is a *create* button which sends an *http.post* to the database.

4.6.2. Mobile

Using the *Ionic* plugin of *MockFlow.com* we obtain the following sketches.



In **Figure 4.6.** we can see the Games Home. In this case we only have questionnaires. We can see two different groups: one group are the active games, which must be the only questionnaires game that can be answered by the student, and the other group is the finished games, which finish date must be passed away. In this case, the page should make an *http.get* to request for all the *questionnaireGame* of the student, and decide which are active and which have been finished.

Figure 4.6. Game Home Sketch

In **Figure 4.7.** we can see an alert dialog over the Game Home Sketch. In this case, this alert dialog should appear when the student click on a finished questionnaire. In this case, the student is not allowed to answer it because the finish date has passed away. Then, an information alert dialog will remind it to the student.

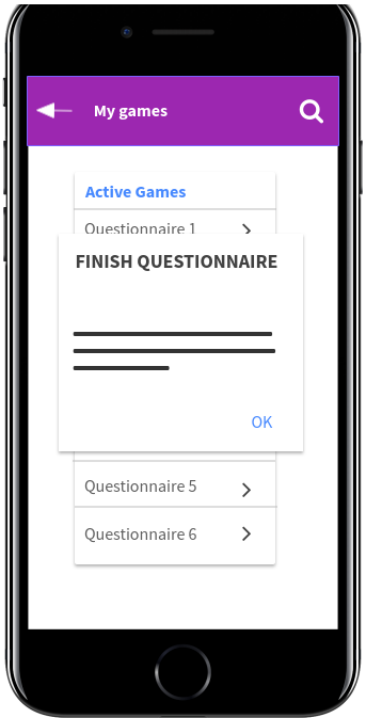


Figure 4.7. AlerDialog of a Finish Game

In **Figure 4.8.** we can see a type of game designed. In this case we can see the game mode 1by1. On the top of the screen we can see the name of the questionnaire, the question time that has the student for each question and on the corner we have the actual number respect to the total number. Below it, we have the statement of the question, a picture in order to help, and all the possible answer to select. In this case there is only one possible answer.

Figure 4.8. 1by1 Page Sketch

In **Figure 4.9.** we can see a type of game designed. In this case we can see the QuizPip. On the top we can see the questionnaire name, and the questionnaire time that has the student to answer the whole questionnaire. There we can see that all the questions are shown toghether, since we can see the answer 5 of the question 1, the question 2 which consits on a open question with a textarea and finally th statement of the next question. If we could scroll down, we will finally get a *submit* button to send the questionnaire.

Figure 4.9. QuizPip Page Sketch

In **Figure 4.10.** we can see another type of game. In this case, we can see the FlipCardsPip game mode. On the top of the picture there is the title of the questionnaire. Below we can see only the statement of the question and the image. If the student want to know the answer, they can go over the statement and the answer will appear. The main purpose is to study. Finally there is a *submit* button that asks for the next question.

Figure 4.10. FlipCardsPip Page Sketch

In **Figure 4.11.** we can see the results page. It appears after each questionnaire is completed, with the purpose of telling the student how the questionnaire was. There is some relevant information on the screen: the final grade obtined by the student, the number of correct and wrong answers and all the answers that the student make.

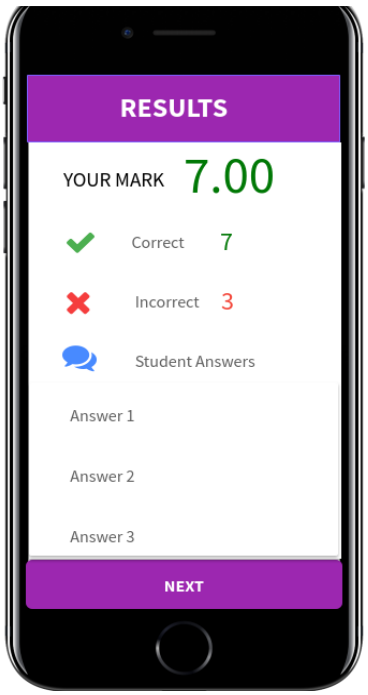


Figure 4.11. Results Page Sketch

Now that we have all the pieces design, it is time to develop our questionnaires games, explaining how they are implemented.

CHAPTER 5. DEVELOP OF THE QUESTIONNAIRE MODULE

Now is time to go through the development of the project and explain how I make the previous design possible. In this chapter I will start from the beginning. First of all we will go through the database, then through the Dashboard and finally to the Mobile implementation. Even so, the first step was the disconnection of the actual implementation of the questionnaires, editing and deleting all the connections of the questionnaire code to the other modules.

5.1. Database

The database design is within the *Classpip-services* repository. This repository is responsible of store all the data and initializing some data to make the application work. For example, creating the users in order to login, creating some test data to prove that the model works, creating some example question…

To see a full explanation of the database with all the fields of each database and their properties you can go to the **Appendix 1**.

5.2. Dashboard and Mobile

The dashboard side is store on the *Classpip-dashboard* repository. This code is focus on the website application.

The mobile side is store on the *Classpip-mobile* repository. This code focuses on the mobile application.

To develop all the different code parts, I follow the next steps. The steps are quite similar between both repositories. The main different are the routes of the components, and for the html structure, Dashboard uses *Angular Material* while Mobile uses *Ionic Framework*.

5.2.1. Create the model

To run the application, you will need to use the database information. Therefore, the first thing you need is a model, to receive all the information that you previously configured in the database. You must create as many as you create in *the Classpip-services*. In this case I will create *Question*, *Questionnaire*, *QuestionnaireGame* and *ResultQuestionnaire*, all of them supported by a .txt file, adding additional information.

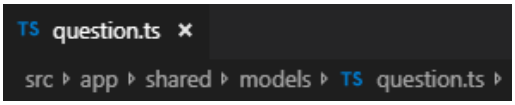


Figure 5.2. Route location of models of the DashBoard

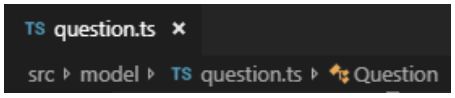


Figure 5.3. Route location of models of the Mobile

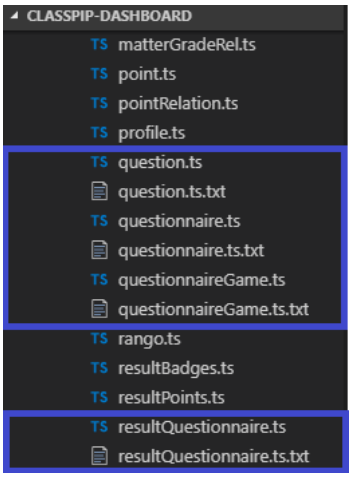


Figure 5.3. Models of the Dashboard

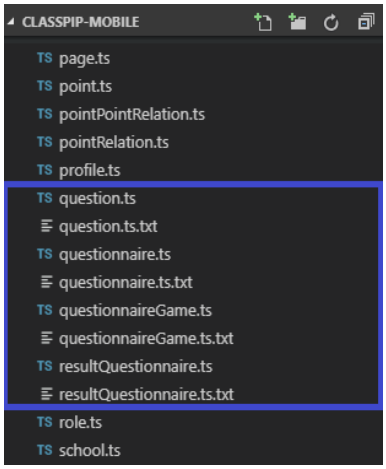


Figure 5.4. Models of the Mobile

In **Figure 5.3.** and **Figure 5.4** we can see a lot of files**:**

Within the typescript (ts) file you can find all the parameters that are in the database. You can also find the constructor (to build de class), *toObject* to build an object of the class and then all the setters and gets to access every parameter of the class

Within the txt file you can find some help about this model, in order to help future developers if they want to modify something.

All this files can be found in the **Appendix 1** in order to have everything together and easy to access and review.

5.2.2. Services

The services module allows all the components of the application to access to the information in the database, such as, get some data, post some information… The purpose is also to share the functions among all of the components and save some code. If you want to obtain some information from two different components you will need to create one function in each of them. Thanks to the services, you can call to the service from both components, and execute the same function.

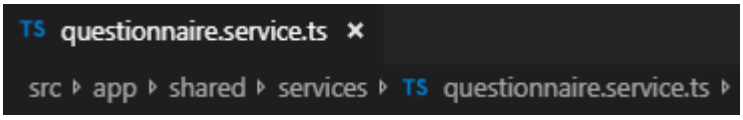


Figure 5.5. Route location of services of the DashBoard

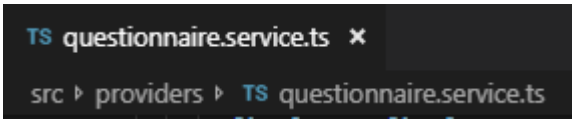


Figure 5.6. Route location of services of the DashBoard

The structure of the services file can be divided into two parts.

> Imports: Here there are declare all the imports that we need in the service. Some imports are needed to access protocols to the database (*http, Response…*), others for the correct behaviour of Angular (*AppConfig, Injectable*…) and the most important ones (*Question, Questionnaire*…) which are related to the models that we have created. We need to introduce the information through the models for allowing all the application work this data. There is also the *ng-2translate service*, which allows the program to be in three different languages. The services module is practically the same as we do similar implementation on both sides. (**Figure 5.7.**)

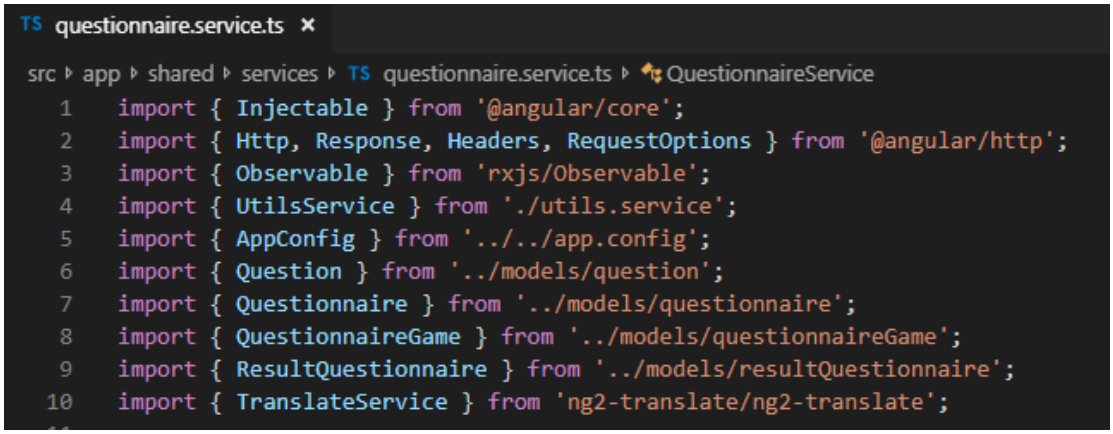


Figure 5.7. Imports of services of DashBoard

> *@Injectable* and functions: The *@Injectable* allows to initialize all the imports that are needed on the module. These functions are all dedicated to make requests (*get*, *post*, *delete*…) to the database. These functions will be accessible for all the program, and could be re-use between them. The way to create this requests, are the same for Dashboard and Mobile, so if you develop something for the Dashboard size, you can easily take the function to the Mobile. (**Figure 5.8.**)

questionnaire.service.ts

src ▸ app ▸ shared ▸ services ▸ questionnaire.service.ts ▸ QuestionnaireService ▸ constructor

```
@Injectable()
export class QuestionnaireService {
  constructor(
    public http: Http,
    public translateService: TranslateService,
    public utilsService: UtilsService) {
  }

  /**
   * Returns the questionnaires of the teacher
   * @return {Array<Questionnaire>}
   */
  public getTeacherQuestionnaires(idTeacher: string): Observable<Questionnaire[]> {
    const options = this.utilsService.getOptions();
    const url: string = AppConfig.TEACHER_URL + '/' + idTeacher + AppConfig.QUESTIONNAIRES_URL;
    return this.http.get(url, options)
      .map((response: Response, index: number) => Questionnaire.toObjectArray(response.json()))
      .catch((error: Response) => this.utilsService.handleAPIError(error));
  }
```

Figure 5.8. GamesResult component of the Dashboard

As we can see in **Figure 5.8.** we have one function called *getTeacherQuestionanires (idTeacher: string)*. As its name says, we can use this service to get the teacher's questionnaires, providing the teacher identifier (*id*). If we go deeper into the function, we can see that it does an *httt.get* to the *URL* of the database, to get the correct information. After that, it will return an array. We can know it thanks to the comment that is above the function *(@return)* or in the *.map response*, that in the end we have a *.toObjetArray*.

5.2.3. Components

The components of angular are divided into four different files that together form the angular component.

Figure 5.9. Components route location of the DashBoard

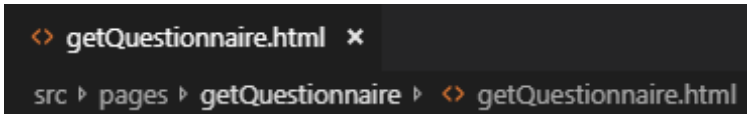


Figure 5.10. Component route location of the Mobile

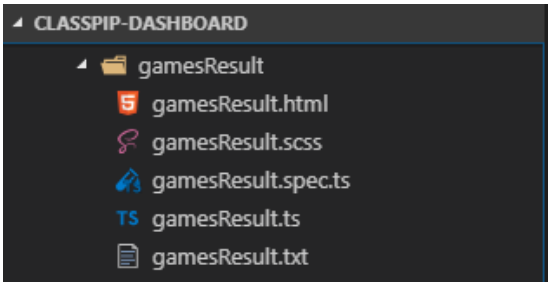


Figure 5.11. GamesResult component

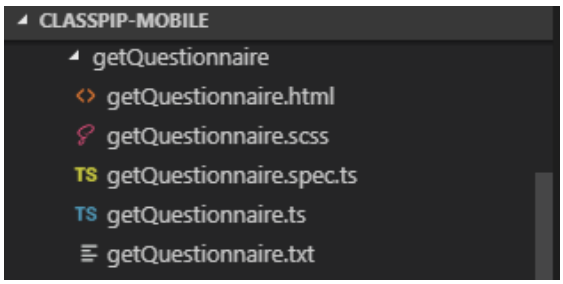


Figure 5.12. getQuestionnaire component

In **Figure 5.11.** and **Figure 5.12.** we can see the MVC (Model – View – Controller) architectural pattern. This is done to separate internal code use, the ways that the information is represented and how the user interact with the component.

With the exception of the html, files with the same extension in both repositories, can be implemented in the same way, re-using the code. Then I will only explain the differences in the html, while for the other files I will only explain one of each.

- gamesResult.html: It is the view part of the architecture. It is mostly based on *Angular Material* using some features of *Angular* like **ngIf* or **ngFor*. (**Figure 5.13.**)

Figure 5.13. GamesResult.html using *Angular Material* of the *Dashboard*

In **Figure 5.13.** we have features of *Angular Material*, *Angular* and *CSS.* These are:

- *<mat-select>:* This is part of *Angular Material.* It allows us to create selects easily with a pre-defined style.
- *{{'GAMES.SELECTQUESTIONNAIRE' | translate}}:* This have access to the translate service which is located on *src/assets/i18n* folder. It allows us to have the application in more than one language. In our case in Spanish, Catalan and English.
- **ngFor*: This is part of *Angular*. This allows us to iterate over the *questionnairesGame* array (note the plural to know that is an array). This will iterate over each position, and the *{{quest.name}}* will print the name.
- Style: in this case, the *CSS* is inside the html tags. It can be added to the *.scss* file.

> getQuestionnaire.html: It is the view part of the architecture. It is mostly based on *Ionic Framework* using some features of *Angular* like *(click)* or **ngFor*. (**Figure 5.14.**)

Figure 5.14. getQuestionnaire.html using Ionic *Framework* of the *Mobile*

In **Figure 5.14.** we have features of *Ionic*, *Angular* and *CSS.* These are:

- <ion-header>: This is part of *Ionic.* It allows us to create selects easily with a pre-defined style.
- (Click): This is part of *Angular*. It allows to call a function when you click the ion-item. In this case, call the function *getQuestionnaire(…)* which will print *{{quest.name}}* about the questionnaire that you have clicked.

> gamesResult.scss: Is the In this case the scss is an evolution of the *CSS*. It allows you to create variables (*$colour: white*) and use this *$colour* everywhere you need in the *scss*. *Scss* means: *Sass CSS*, which means Syntactically Awesome Stylesheets (*Sass*) Cascading Style Sheet (*CSS*). (**Figure 5.15.**)

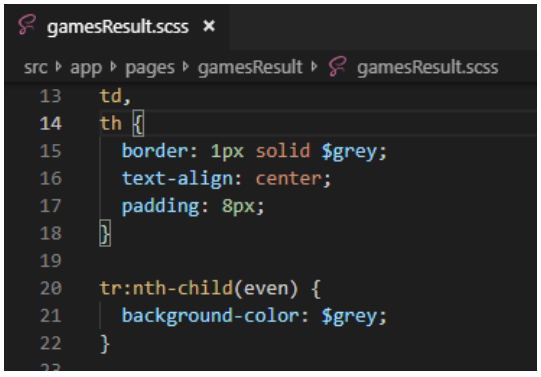


Figure 5.15. GamesResult.scss of the Dashboard

In **Figure 5.15.** we can see the *$grey* attribute that is repeated twice. This *$grey* is declare at the top as: *$grey*: *#dddddd*, which *#dddddd* is the

hexadecimal code for the grey colour. In this case we can avoid writing the hexadecimal code if we use the variable. It is only possible thanks to the evolution of *CSS* to *SCSS*.

> gamesResult.spec.ts: This is used for tests. It is usually recommended to do some tests of the component and test all the possibilities that could occur during the use of the application. You can test everything you need, such as the declaration of the variables (expected input) and the final results after using the variables (expected output). (**Figure 5.16.**)

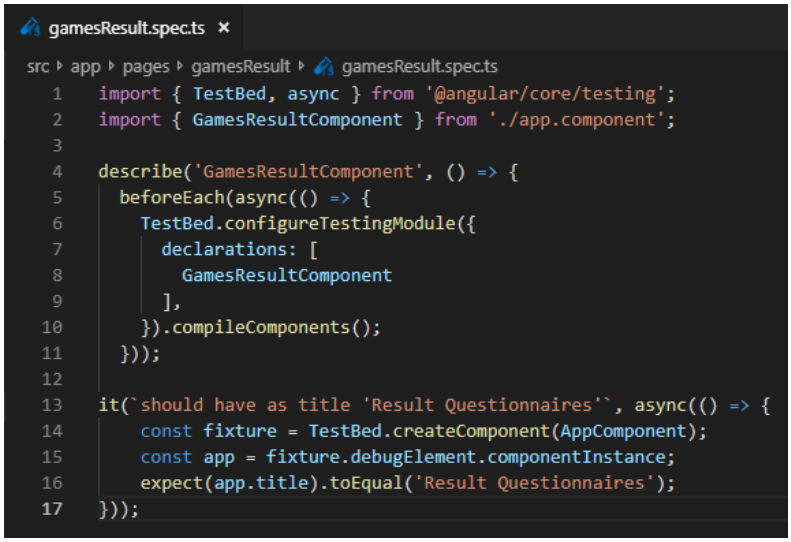


Figure 5.16. Test of gamesResult.spec.ts on the Dashboard

In **Figure 5.16.** we can see the test that we have done to the gamesResult component. Normally the test have two parts:

- Describe: where we declare and import our component to test.
- It: Where we test some features. In this case, we are testing if the title is equal to “Result Questionnaires”. We can add as many it as we want. The program will launch one respond for each test.

> gamesResult.ts: Is the controller part of the architecture pattern. It is the core, the most important part of each component. Here is where the information is treated and decide which will the next step of the application. They are divided in 3 parts:

- Imports: Always located at the top of the file. Its utility is to import all the classes, services, components… that we need to treat the information correctly. (**Figure 5.17.**)

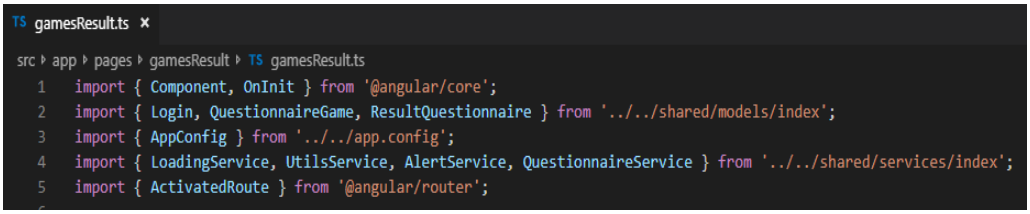


Figure 5.17. Imports of gamesResult.ts on the Dashboard

In **Figure 5.17.** we can see all the different imports we have. As we can see, there is an @angular import, which is related with internal characteristics of the same. We have also one import for the models (*QuestionnaireGame*, *Login*…) to get access to the classes and manage the information. Finally, we also import the services (*QuestionnaireService*, *LoadingService*…) to access the database or some other services that we might need, such the *LoadingService* to show a loading bar to the user.

- Component. The most important part of the file .ts. Here is declared which other files (.html, .sccs…) will use the *Angular Component*. It is also declared all the variables that will be used in the component and the constructor to obtain access to imports. (**Figure 5.18.**)

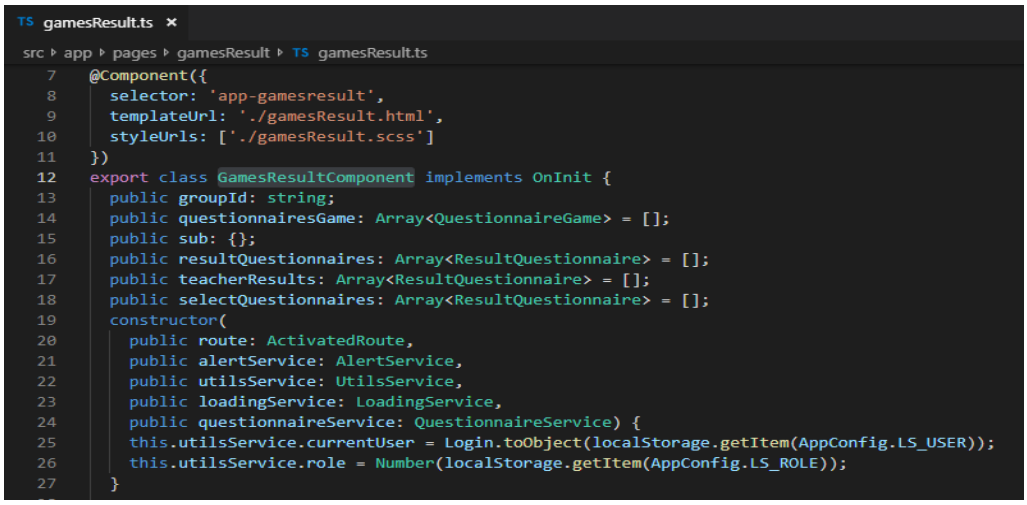


Figure 5.18. Component of gamesResult.ts of the Dashboard

In **Figure 5.18.** we can see above the @Component which the declarations of the files, that should the names that appeared in **Figure 5.11.** . After that we have the several declarations of the variables, some are strings and some are arrays and finally the constructor with the imports.

- Functions: This is the final part of the file, where you can create as many functions as you need. Here you will receive all the information that the users input to the system (using the html) and you will treat it correctly. (**Figure 5.19.**)

Figure 5.19. Functions of gamesResult.ts on the Dashboard

In **Figure 5.19.**, we can observe two different types of functions:

- NgOnInit: This is the first function that is fired when the program enters to the controller. It is even fired before the html shows the information. It is used to get all the data that will be needed in the component (including *html*). Then it will launch two functions in order to get access to the database data.
- GetResultsOfQuestionnaires: This is one of the many functions that the program has. In this case is use to obtain the results of the questionnaires, thanks to the service *QuestionnaireService* and the function *getRestults* explained above. It will also verify that the result belongs to the teacher and after that the result will be added to an array.

> gamesResult.txt: This files are some additional information I give to the future developer to help if they need to modify something about my code.

The previous component, is one of the many components that this project has been added to the Classpip project. It is an example of how the code was develop. If you want to know all the other components you can go to the Appendix 2.

5.3. Clean Code

This project is far from be individual. That is why all developers should keep in mind that this code is not yet finished, that someone else will come to extend this application a little bit more. These extensions can affect your code, so it is advisable to leave the code as clean as possible, as well as the functions and variables.

Knowing this, there are many techniques on the Internet to make our code clean and understandable. By reading several of them, I have reach to a work methodology that I believe is easy to use and understand, and that can allow the future developer to modify my code easily.

Then, along the Dashboard and Mobile code, you can find these type of variables:

- *Title: string;* Variables of type string will always be in singular.
- *questionnairesGame: Array<QuestionnaireGame> = []; Arrays* will always in plural (to remark that there will be more than one) and will be initialized in the constructor
- *NumMark: number;* Variables of type *number* will always start with num, unless it is easy to identify that this variable is a *number*.
- *FindCategory: Boolean=false;* Variables of type *Boolean* will always start with *find* due to the fact that they are normally used to find something while iterating an *array*, unless it is easy to identify that this variable is a *Boolean*.

In case of the functions and the services, you can find:

- *Public getQuestion (idQuestion: number):* All functions will be clear and easy to understand. Most of the functions use the command syntax to access a database: *Delete*, *get*, *post*, *patch*… Thanks to that, it is easy to use the correct function thinking about what you need. It is also clear what you need to put into the function in order to help the developer.

5.4. Final results

Once all the development of the project has been done, it is time to evaluate the results of the application. It is time to think if I achieve all the previous design. It is also time to give the application to external people and see how they manage to use it. This will give us some possible improvements.

This is how the application at the end looks.

5.4.1. Dashboard

Following we can observe some screenshots of the Dashboard application.

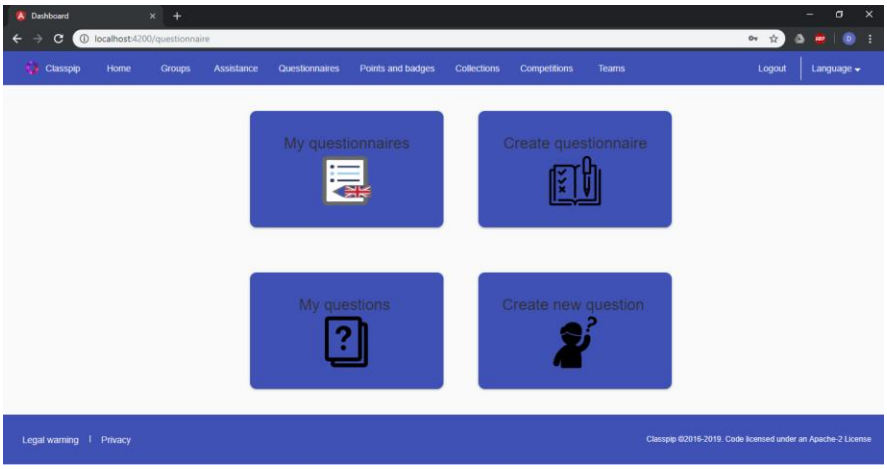


Figure 5.20. Questionnaire home page

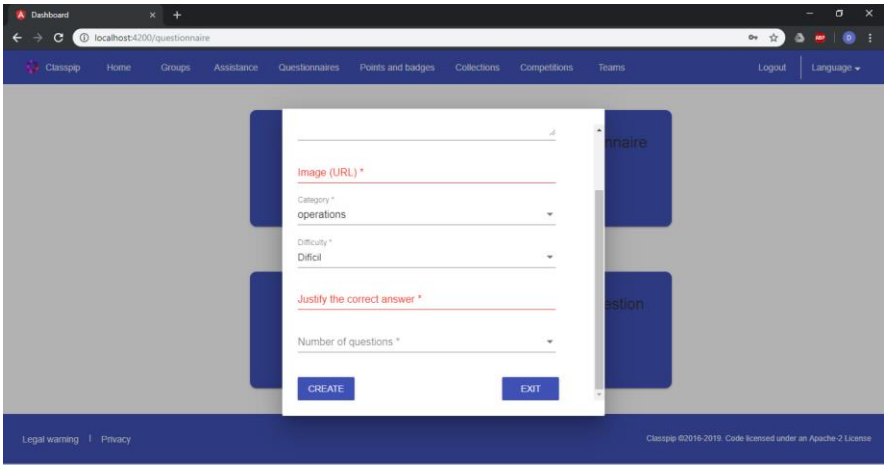


Figure 5.21. Creating a question page

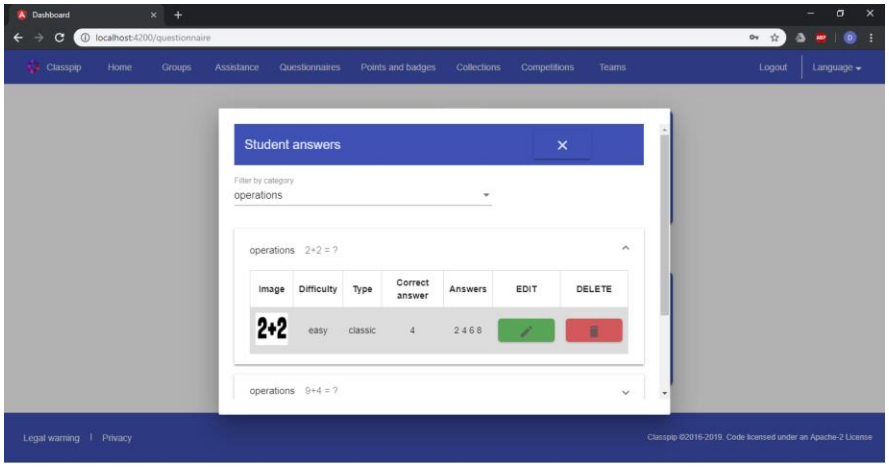


Figure 5.22. Creating a question page

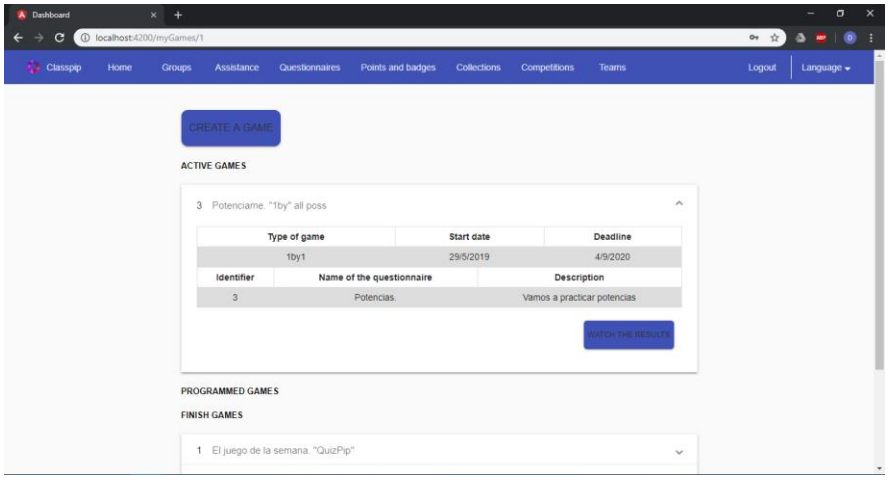


Figure 5.23. Game home page

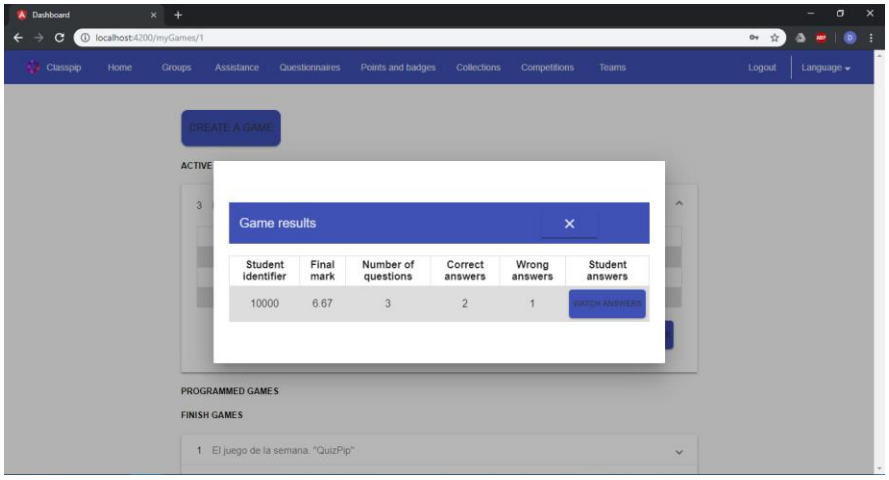


Figure 5.24. Game results page

5.4.2. Mobile

Following we can observe some screenshots of the Mobile application.

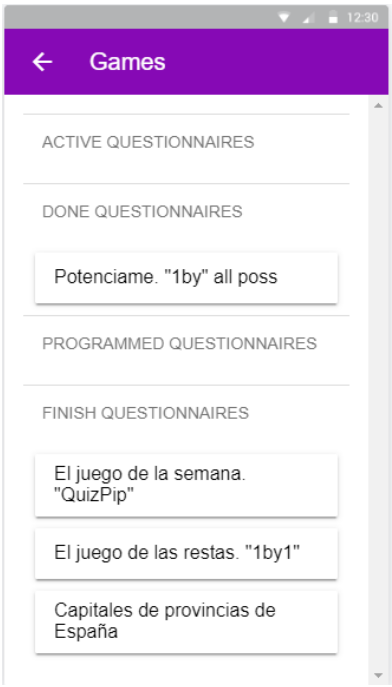


Figure 5.25. Games home page

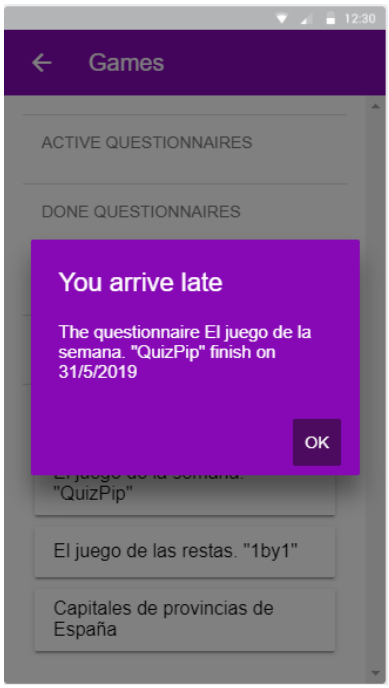


Figure 5.26. Alert dialog game

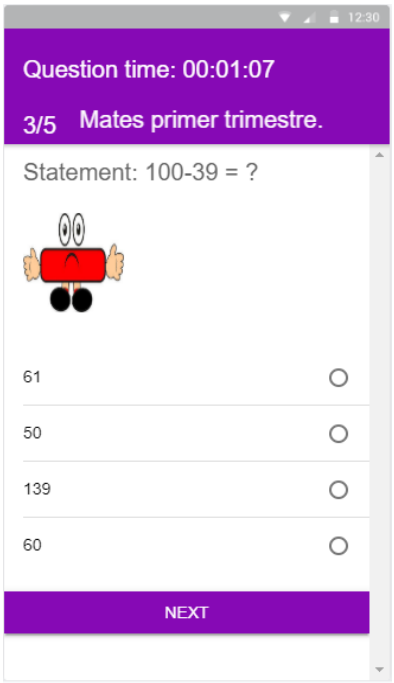


Figure 5.27. 1by1 game mode

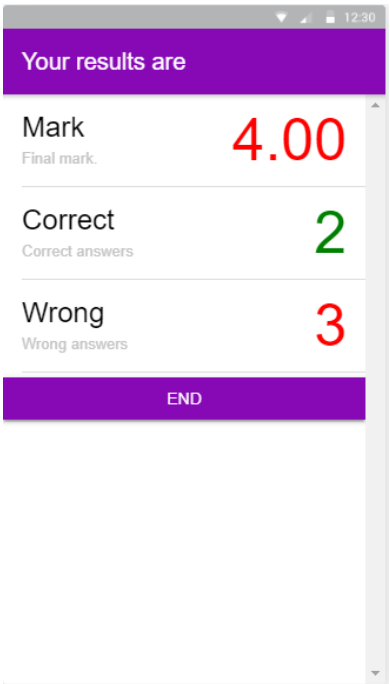


Figure 5.28. Result page

CHAPTER 6. TESTS AND ASSESSMENT OF THE MODULE

Now that the module of questionnaire within the Classpip Repository is finished, it is time to perform tests of it in order to assure that the program is robust and meet all the initial design an expectations. This tests are the evidence that today, when I deliver the project, everything works as expected. These tests can be perform by future developers that modify my modules, with the purpose of verify if the modules work in the same way as it was originally develop.

When you develop something on your own without any help, you may think that the application is clearer than it really is. Maybe it lacks relevant information and the future user is lost in some point of the program. That is why submit my program to an external evaluation in order to improve some parts of the application.

6.1. Tests

We could differentiate between tests on the Dashboard or on the Mobile.

6.1.1. Dashboard

On the dashboard size I can ensure that:

- When you want to create a Question, a Questionnaire or a Game, all the input fields and selector must be fulfil if they are required. If you do not accomplish that, you will get a toast message on the bottom reminding you that "*All fields must be filled correctly*".
- When you create a Question, a Questionnaire or a Game is store correctly on the database. You can now that because you can see what you created instantly.
- When you have to introduce a much longer text, there is a text are to help the teacher to visualize all the field. Then, when this text needs to be print, the label is readjust. This case is the statement of the question.
- When you create a game, the game will be placed in his position. Depending on which start and finish date you introduce, it will go to programmed questionnaire games (the start date it has not passed yet), active questionnaire games (the finish date it has not passed yet) or finished questionnaire games (the finish date it has passed).
- The results button of a questionnaire game will only appear when there are results of that questionnaire. If not, the button will remain hide.
- When you create a questionnaire, you can select as many question as you want.

6.1.2. Mobile

On the mobile size I can ensure that:

- The student can only access to the active questionnaire. If they tried to access one of other type, there will be an alert dialog showing some useful information.
- The tree type of games already implemented (Quizpip, 1by1 and FlipCardsPip) can be played without any problem, using or not the time configuration.
- The results of each game (except FlipCardsPip), are stored on the database and the teacher will have access on the Dashboard. You can see the results clicking on the “view results” button on one questionnaire game.
- The student can only play once each game (except FlipCardsPip). If you tried to play again, an alert dialog will appear.
- The marks of the questionnaire will appear after answering the questionnaire. In the case of FlipCardsPip, as it is focused to study, there are not any marks.

6.2. Assessments

Once I consider my application is finished, I let several people try my application and receive some feedback from them. A total of four people try the original application. This allowed me to verify that my application is friendly and easy to use. As expected, there were many details that I overlooked because I knew at all times what I had to do in each place of the application, since I am the one who designed and implemented everything. These are all the points which my application was not entirely satisfactory. Most of the points were about the user experience using the application or how were designed the screens.

The feedback I received and I have could improve are:

- On the Dashboard side:
  - All the *Dialogs* (screens that appears above our actual screen and they do not occupy the entire screen) I create, did not have any *exit button*, as I am familiarize to click over the grey background that surrounds it to close the dialog. Probably a lot of people would not understand that at a first sight, and they will feel lost.
  - Teachers could not filter *Questions* by *category*. It would be useful to filter the *questions* to see them more comfortably.
  - Teachers could not select *categories*. They could just add it, remembering all the categories they made. These, would create some discomfort as there could be the same category with little differences: “maths, Maths, mathematics….”

- The D*ialogs* were not responsive. For a larger/smaller screen, where not shown in the middle, which does not make the user have the desired experience.
- When the *Question*, *Questionnaire* or *QuestionnaireGame* information was shown, a lot of irrelevant information was shown. It was useless information, which may not be necessary.
- When the teacher was creating a new *QuestionnaireGame*, they could select what kind of *game*, but only the name was shown. Probably the teacher could forget what the game was about.
- In the *ResultQuestionnaire* table could be useless to show all the answers in the same table, because if the questionnaire has many *Questions*, the table will grow inefficiently. It will be good to implement a filter.
- Teachers cannot *modify* or *delete* any *Question*, *Questionnaire* or *QuestionnaireGame*. These options allow the teacher to do a good maintenance of the application. I only could to implement *modify* feature.

❖ On the mobile size:

- When answering an *open answer Question*, the response area, which was implemented by a *text area*, was unclear. As it does not have any edge and any placeholder, it was hard to guess where to write the answer.
- The student could fulfil the *Questionnaire* as many times as they want. It was not efficiency, because they would repeat until they get the perfect mark.
- *Alert Dialogs* were only available in *Catalan*. There were not translated into *Spanish* or *English*.
- The students do not have any reference in which *Question* they are, and how many *Questions* are remain. That makes that the *Questionnaire time* cannot be managed correctly.

The improvements that I could not make are:

- Teachers cannot *delete* or *modify* any *Question*, *Questionnaire* or *questionnaireGame.* Loopback framework does not *delete* all the references from the others tables to the register that you deleted. That makes that if I want to *delete* a *Question*, I need to search into each *Questionnaire* and delete the *question identifier.* If I want to *delete* a *Questionnaire* the same. But in this case, as a *QuestionnaireGame* has one *Questionnaire*, if I *delete* the *questionnaire* I should probably also *delete* the *QuestionnaireGame* because then the game will be empty*,* and probably the *ResultQuestionnaire* because the *questionnaireGame* disappeared.

CHAPTER 7. ONBOARDING MATERIALS

As our project is part of a large project, you have to ensure that other people can easily understand your code. That is why it is recommended to give some advices to the following developers, giving some clues about your code. Also, as you become an expert in the code, since you have been working on it since several months, you can do an easy tutorial to them, which shows how to develop something simple.

7.1. Tutorial 1: Walk around the code

As the code at first glance can be somewhat difficult to understand, since there are a lot of repositories, folders and code lines, it would be good to have some code tutorial to explain everything that have been developed.

In this tutorial, you will learn how the code of the questionnaire is structured and try to understand the whole reason why it was done in that way.

Links to YouTube platform:

- Services: https://youtu.be/ep65yAI70uI
- Dashboard: https://youtu.be/UDucKug_glM
- Mobile: https://youtu.be/eUiRBUnqPOo

7.2. Tutorial 2: How to implement something

Up to this moment of the project only tutorial of learning has been realized. This tutorial is called "*Tutorial Mesa*" and was develop in a primitive Classpip where there were few modules implemented. This tutorial is done in written mode which takes a bit to get lost sometimes where the parts of the code are. That is why I want to implement something simple but from the beginning to the end, to help new developers in their first steps.

In this tutorial we are going to create a pet model. This pet model will be related with Students in a relation of 1:1 ratio (one pet for each student, with no possibility to share it). Students will have the possibility of check their pet from Dashboard and Mobile. This implementation will take us through all the steps I take in **Chapter 4** of this document.

Link to YouTube platform: https://youtu.be/wmDC4Hdoaxw

Links to GitHub website:

- Services: https://github.com/erdeivit/PetTutorial-Services
- Dashboard: https://github.com/erdeivit/PetTutorial-DashBoard

CHAPTER 8. CONCLUSIONS

The conclusions are divided into different parts, from the most technical point of view to the most personal, going through the challenges faced up during the development of the project.

8.1. Technical

This project could be considered a highly technical one, not in terms of creating any device, in terms of risk of failure. This project combines several technologies, most of them new to me, old code develop by another person and requirements of great demand by teachers, which creates a dangerous mix.

8.1.1. Framework and tools

To develop a code project, is important to correctly define the technologies to be used. In this case, these two technologies were predefined.

For the dashboard we use *Angular. Angular* is a powerful framework that allows you to create applications easily, both as websites as mobile applications. In the Dashboard size, we get the support of *Angular Material* to create a better user interface, thanks to the amount of predefined components it has available on the website. For the Mobile we use *Ionic,* which helps you to make the user interface for mobile. It has many advantages, since it easy to implement with *Cordova*, and its application will be compatible with iOS and android.

Another important tool is git and *GitHub*. In this case, I did not use all the possibilities it offers us. In this case, I use more git to create local copies of the code, in order to go back if something goes wrong. But if you are in a collaborative development, you can upload a copy each time to *GitHub* an all the team can downloaded it easily.

The last technology is *Loopback.* This is uses to store our local database. It has many advantages, for example, you can perform tests of the functions in the web interface. The main disadvantage, is that when you want to delete something, you need to go model by model deleting all the relationships. Other technologies do it automatically.

8.1.2. Faced up challenges

The main challenge is to achieve what you committed to your advisor. Despite all the possible problems I suffer during the development process I was able to obtain a majority point that my advisor is very satisfied with what we can do now with my modules, compared with the old implementation of the questionnaires module. Also is important, to not lengthen in time, which means that you have a project deadline, which cannot be moved widely.

There are also other challenges when you develop a project. In this case I could highlight two important ones:

The first, like all the developers, needs to become familiar with the technology and the code. In my case, I did not use any of these technology before, so I had to understand it while I was using it. There are a lot of code lines, with some of which you need to interact. This makes that the first steps are quite slow. In addition, you need to get everything fast if you want to accomplish the deadlines of the project.

The other is work with dates. So far, any of the previous developers' uses dates and times. All of them, simply store it in the database but none of them uses that information. Working with dates is a real challenge, because you need to format it, and make calculations with it in a special way.

8.2. Personal opinion

I am very satisfied with the results I obtained in this project. I am very proud of what I develop and the level that it reaches. I am also glad to learn these emerging and powerful technologies, which can give a future job if I decide to be a website programmer using these web technologies. I fulfil the desire that I have which was explained on the personal motivation (**See Chapter 1.3**), and I can say I feel confident creating a high qualified code.

Also this topic of gamification, attracts my attention because I like children and I will probably work with technology and children. That is one of the main reasons I decided to join this project, since I did not know this concept of gamification.

As technologies, I prefer using *Angular Material* than *Ionic*. *Angular Material* has better pre-defined features with more much information and lot of examples, while *Ionic* you need to search deeper on the website.

Face this big project by my own, it has been a great opportunity to develop some personal skills, gives me the opportunity to demonstrate the ability to learn and adapt to new challenges. Also the fact that face the problems by my own, knowing that I have to solve every problem appears, it creates a bit of anxiety but at the same time wanting to fix it.

The fact that this code will be part of something bigger, and it will not be stopped once I finish (like many lonely TFG) make me the motivation to finish the code in a proper way.

8.3. Future improvements

Probably the most critical point of my entire project, is that it lack of connections with other modules. It lack of connectivity and functionalities connected to the other modules of Classpip. That give us, several ways to improve the project.

- As it was mention on **Chapter 6.2.** teachers cannot delete and modify *Questions*, *Questionnaires* and *QuestionnairesGame*.
- Connections with competitions. Decide which is the winner of the journey, depending on the final results of one questionnaire.
- Finish the designed games that were describe on the chapter 3.
- Improve the *CSS* of all the questionnaire modules.
- The way of create a questionnaire. In this case I used a multiple selector to decide which *Questions* will belong to the *Questionnaire*. If the *Question* list start go grow up, the selector will be so big.

CHAPTER 9. APPENDIX

Appendix 1: Structure of the database

I am going to explain deeply the structure of our database. First of all we have an extended flow diagram will all the fields of each database. This flow diagram it was made using the free website https://creately.com/app/# which allows you to easily create database diagrams.

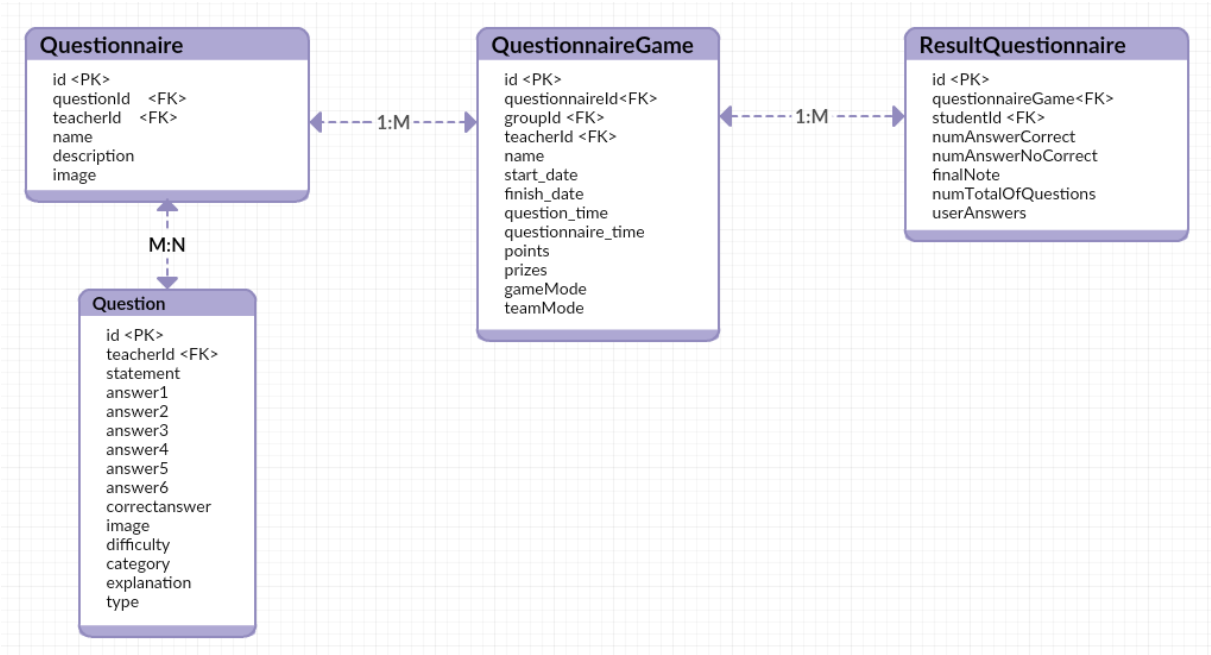


Figure 9.1. Database fields and relationships

The **Figure 9.1.** is accessible on:
https://creately.com/diagram/jws4jveu3/BD7ViuKnh1rnJ63XQIzNeM2WRTE%3D

Now I am going to describe each model which configuration has, in terms of the Classpip-services repository.

1.1.1. Question

In **Figure 9.2.** we can have a look of the properties of the model *QuestionnaireGame.* Also there is the relations with the others databases, which theirs *foreingKey*.

```
"properties": {
    "statement": {"type": "string","required": true,"datatype": "longtext"},
    "answer1": {"type": "string","required": true},
    "answer2": {"type": "string","required": false},
    "answer3": {"type": "string","required": false},
    "answer4": {"type": "string","required": false},
    "answer5": {"type": "string","required": false},
    "answer6": {"type": "string","required": false},
    "correctanswer": {"type": "string","required": true},
    "image": {"type": "string","required": false,"datatype": "longtext"},
    "difficulty": {"type": "string","required": true},
    "category": {"type": "string","required": true},
    "explanation": {"type": "string","required": true},
    "type": {"type": "string","required": true}
},
"relations": {
    "questionnaire": {"type": "hasAndBelongsToMany","model": "Questionnaire",
      "foreignKey": "questionnaireId"},
    "teacher": {"type": "belongsTo","model": "Teacher",
      "foreignKey": "teacherId"}
}
```

Figure 9.2. Question propierties and relations

With this file, there is a *txt* file (*question.txt*) which include some extra information about some fields, in order to know all possible values that may have according to all the possible configurations. The most special ones are:

- *difficulty*: It can take these values: *hard*, *medium* or *easy*.
- type: It can take these possible values:
  - *classic*: The question only has one correct answer.
  - *multiAnswer*: The question has more than one correct answer.
  - *openQuestion*: The question has no possible pre-defined answers, and students will answer using a text area.

1.1.2. Questionnaire

In **Figure 9.3.** we can have a look of the properties of the model *Questionnaire*. Also there is the relations with the others databases, which theirs *foreingKey*.

```
"properties": {
    "name": {"type": "string","required": true},
    "description": {"type": "string","required": false},
    "image": {"type": "string","required": false}
  },
"relations": {
    "question": {"type": "hasAndBelongsToMany","model": "Question",
      "foreignKey": "questionnaireId"},
    "questionnaireGame": {"type": "hasAndBelongsToMany",
      "model": "QuestionnaireGame","foreignKey": "questionnaireId"},
    "teacher": {"type": "belongsTo","model": "Teacher",
      "foreignKey": "teacherId"}
  },
```

Figure 9.3. Questionnaire propierties and relations

With this file, there is a *txt* file (questionnaire.txt) which include some extra information about some fields, in order to know all possible values that may have according to all the possible configurations.

1.1.3. QuestionnaireGame

In **Figure 9.4.** we can see the properties of the model *QuestionnaireGame*. Also there is the relations with the others databases, which theirs *foreingKey*.

```
"properties": {
    "name": {"type": "string","required": true},
    "start_date": {"type": "number","required": true},
    "finish_date": {"type": "number","required": true},
    "question_time": {"type": "number","required": false},
    "questionnaire_time": {"type": "number","required": false},
    "points": {"type": "array","required": false},
    "prizes": {"type": "array","required": false},
    "gameMode": {"type": "string","required": true},
    "teamMode": {"type": "number","required": true}
},
"relations": {
    "group": {"type": "hasAndBelongsToMany","model": "Group",
      "foreignKey": "groupId"},
    "teacher": {"type": "belongsTo","model": "Teacher",
```

```
      "foreignKey":"teacherId"},
    "questionnaire": {"type": "BelongsToMany",
      "model": "Questionnaire","foreignKey": "questionnaireId"}
}
```

Figure 9.4. QuestionnaireGame propierties and relations

With this file, there is a *txt* file (*questionnaireGame.txt*) which include some extra information about some fields, in order to know all possible values that may have according to all the possible configurations. The most special ones are:

- *start_date* and *finish_date*: This store the start date or the finish date respectively, of the *questionnaireGame*. Store the equivalent of the *Date* in *number* (The number of milliseconds that has passed from the 01/01/1970 day).
- *question_time* and *questionnaire_time*: This store the maximum time that has the student for answer each *Question* or the *Questionnaire*, respectively. Is stored in seconds.
- *gameMode*: This store the type of games of the *questionnaireGame*.
  - *Quizpip* (Already implemented): Classical way to answer a *Questionnaire*. All the *Questions* will be displayed on the screen. Students will press submit button unless the time pass away.
  - *1by1* (Already implemented): On the screen will appear one *Question* each time. Students answer the *Questions* and when they press the submit button (or the time passed away) will appear the next. Question
  - *FlipCardsPip* (Already implemented): This mode is focused on the study. The *Question* will appear on the screen and when students think that they guess the *Question*, they will be able to know the *answer* (passing the mouse over there).
  - *Qlasspip* (Not implemented): It is thought to be similar to *Kahoot*.
  - *TimeToLearn* (Not implemented): It is thought to launch a *Question* (through bar notification) which will give points to the student. It can appear at any time (8:00-22:00).
  - *TrivialClip* (Not implemented): It is thought to be similar to a trivial game.

1.1.4. ResultQuestionnaire

In **Figure 9.5.** we can see the properties of the model *ResultQuestionnaire*. Also there is the relations with the others databases, which their *foreingKey*.

```
"properties": {
    "questionnaireGame": {"type": "QuestionnaireGame","required": true},
    "numAnswerCorrect": {"type": "number","required": true},
    "numAnswerNoCorrect": {"type": "number","required": true},
    "finalNote": {"type": "number","required": true},
    "numTotalOfQuestions": {"type": "number","required": true},
    "userAnswers": {"type": "array","required": true}
  },
  "relations": {
    "student": {"type": "belongsTo","model": "Student",
      "foreignKey": "studentId"}
  },
```

Figure 9.5. ResultQuestionnaire propierties and relations

With this file, there is a *txt* file (*resultQuestionnaire.txt*) which include some extra information about some fields, in order to know all possible values that may have according to all the possible configurations.

Appendix 2: Code in GitHub

The code of the application can be find in my repository of GitHub on the following links.

Services: https://github.com/erdeivit/Classpip-Services

Dashboard: https://github.com/erdeivit/Classpip-Dashboard

Mobile: https://github.com/erdeivit/Classpip-Mobile

To download the code you need to go to a *cmd* go to some specific carpet and execute:

git clone **repository**

where repository is one of the above links

Then execute:

npm install

For run the Services repository:

npm run start or **node .** (dot include)

For run the Dashboard repository:

npm run start or **ng serve**

For run the Mobile repository:

Ionic serve or **ionic serve - - lab** (to run it in mobile view)